Background:

Your friend Rachel told you that she's having a new gig - she will run a sandwich stand! Any of her customer could order a sandwich, and each morning Rachel make all the sandwiches at home, and when she arrives at the campus she pings all the customers.

Thing is, Rachel doesn't know Android development and she needs your help to write an Android app for her customers.

Instructions:

In this exercise you will learn:

1. Work with a remote DB using firebase's firestore
2. Handle your own UI/UX given a needed spec

Firestore Specs:

You and Rachel agreed that every sandwich order will have the following fields:

id: (string) some random-generated string which will be the order's id
customer-name: (string) name of the person who ordered the sandwich (so Rachel could know who to give it to)
pickles: (int) amount of pickles in the sandwich. in the range [0,10]
hummus: (boolean) should we add hummus
tahini: (boolean) should we add tahini
comment: (string) comments from the customer if they want to write anything
status: (string) status of the order. could be one of "waiting", "in-progress", "ready" and "done". when you upload the order, you upload it with state "waiting". as long as the state is still "waiting" you can edit the order. once Rachel starts working on the order she will change the state to "in-progress" and you can't edit the order anymore. once she is in campus she will change the state to "ready" which means that the customer can come grab the sandwich. once you showed the customer that the sandwich is ready, you change the state to "done" and both you and Rachel can stop looking at this order.

You and Rachel agreed that you will both read & write the sandwich orders to the below collection in firestore:
orders/
in this collection you will add documents. each document is a description of a sandwich order (as described above)

For example, to create a new sandwich order with the random-generated id "a6sf3", you should upload it to "orders/a6sf3", and have the order's "id" field to be "a6sf3".

UX spec:

Rachel wants easy app for her customers. They won't need to look at history or past orders. They don't need any fancy login page. All she wants of them is as follows:

1. A "new order" screen to add a new order. If there is no order currently running, when the customer opens the app they will see this page. Here they can select how many pickles, tahini, hummus and hand-writing comments. a "save" button will let the user confirm the order and you will upload it to firestore, and then you will navigate the user to... -->

2. An "edit your order" screen. As long as the order is still in status "waiting", the user can edit the order. if the customer kills the app and re-launches it, and their order is still in status "waiting", they will get immediately to this screen. in this screen: let the user **edit** any of the order's ingredients & comments. let the user **delete** the order.

3. an "your order is in the making..." screen. when the order status is "in-progress" this will be the only screen that the user will see. in this screen there will be some explanation, like "Rachel is working on your order, she will let you know once it is ready." if you want to add nice enhancements it could be cool. like an animation of a ketchup being poured into a sandwich. or just a loading scrollbar. whatever you want to give the user a feeling of "you have nothing to do but Rachel is working hard on your sandwich"

4. An "your order is ready!" screen. when the order status is "ready" this will be the only screen that the user will see.
in this screen: let the user know that Rachel is in campus with their sandwich. the user will have some way of acknowledge (for example, a "got it" button). once the user acknowledge, we will mark this order as done and will navigate the user to the "new order" screen.

Screens spec:

once there is a running order, you should save the order's id in the phone so you can query firestore to get the order data and its current status. the app will only have 1 screen at any moment (e.g. tapping BACK from a screen will not get to any previous screen).
The full UI navigation is:
- no order / order with status "done": delete the order id from the phone and show screen "add new order"
from this screen user can save a new order with status "waiting'

- order with status "waiting": show "edit order" screen
from this screen user can delete order or update its fields (still same status)

- order with status "in-progress": show "order in the making" screen

- order with status "ready": show "order is ready" screen
from this screen user can set a new status "done" to the order

notification spec:

Rachel would want a notification to be shown to her customer when the order is ready. As a true friend, you told here open heartedly that PostPc is not the single course you're taking this semester and you don't think you can make it. Rachel laughed and said it's ok :) notifications will come one day in version 2.0 or something. (no need to support notifications)

Guidance:

You would want a single source of truth. One instance that will upload&download data from firestore, and all the various screens will talk with that instance. Initiate that instance from the . The current order-id can be stored in SP and passed to the source-of-truth in the constructor.

UX:

The spec is fluid enough to raise some interesting UX points for you to solve. Some examples:

- Each order should have "customer-name". The user doesn't change name from order to order. How will you optimize this field so that the user won't need to re-insert their name each time? will you expose a way to change the customer-name once inserted?

- Pickles. you should get a number from the user. how will you make sure that the pickles amount is always between 0-10? will you put an edit text with guards? show a slider? add a "plus" and "minus" buttons and disable them when needed? another option?

- Texts. You can show generic texts or customize them. For example you can drop a generic "you chose <x> pickles" text and call it a day. But what will the user feel when x==0? it is much less fun then having edge-case text for 0 pickles ("no pickles for me please :)") or 10 pickles ("I want many picklesssssss")

As you can see, when project-complexity rises, the UX starts being interesting!

Tests:

You should have at least 3 unit tests or flow tests. Feel free to add tests. The more the merrier! To add robolectric into your project take a look here. To add mockito take a look here.

Submission:

you should push your work (from your local computer) into a public git repository as you did in previous exercises.

In moodle submission you should submit a url-link to your git repository, located in github, gitlab or huji-cs-git.
Your repository should be **public**. so I can see your content.

When I will go to your repository and look for the latest commit, I should find:

1. this commit has been commit before June 5th, 2021 at 23:59

2. this commit contains all the needed source code to run your app without build errors

3. all unit tests in all test files are passing.

4. this commit contains a README file in the main folder, in which you promise you have written the code alone.
If you're having trouble with Formulation you can use this:

"I pledge the highest level of ethical principles in support of academic excellence. I ensure that all of my work reflects my own abilities and not those of someone else."

Remarks:

work with git. Since this is a do-your-own-thing-from-scratch kinda project, it is very important that I would be able to understand your work flow. Make a commit every time you made progress, so you will end up with a lot of commits and the commit messages will tell the story of how you solved this exercise.

~*~*~*~

Good luck!